Amendments to the Claims

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A runtime system comprising:

a plurality of processors;

a global address space language program comprising a plurality of program threads that access memory in a global address space system;

a directory of shared variables, for locating and managing shared objects, the directory comprising a data structure for tracking shared variable information that is shared by a plurality of program threads and entries acting as object identifiers, providing a level of address translation between a compiler and the runtime system;

wherein any of the plurality of program threads can access a common control data structure and the plurality of program threads include a calling thread that allocates space and inserts a handle in its partition in the directory of shared values; and

allocation and de-allocation routines for allocating and de-allocating shared variable entries in the directory of shared variables;

wherein the runtime system is implemented on a distributed memory system, said runtime system further comprises:

a private memory of each thread, the private memory comprising a replica of the directory of shared variables such that said directory is replicated across all of the threads.

2 - 3. (Canceled).

4. (Original) The runtime system of claim 1 wherein the runtime system is implemented on a shared memory system and the directory of shared variables is stored in a shared memory shared by all threads.

5. (Original) The runtime system of claim 1 wherein the allocation and de-allocation routines are used for both statically and dynamically allocated data.

6. (Original) The runtime system of claim 1 wherein arrays that are dynamically allocated have affinity to a thread that called the allocation or de-allocation routine.

7. (Original) The runtime system of claim 1 wherein every thread has a handle for each shared variable that it accesses.

8. (Original) The runtime system of claim 7 wherein the entries in the directory of shared variables are accessed using the handle.

9. (Original) The runtime system of claim 7 wherein the handle comprises a partition index and a variable index.

10. (Original) The runtime system of claim 1 wherein each thread has exclusive write access rights to a partition of the directory of shared variables associated with the thread.

11. (Currently amended) A runtime system comprising:

a plurality of processors;

a global address space language program comprising a plurality of program threads that access memory in a global address space system;

a shared data directory, for locating and managing shared objects, the directory maintaining shared data entries related to shared data structures that are shared by more than one of the plurality of threads; and

control structures to access, allocate and de-allocate the shared data structures through the shared data directory;

wherein the runtime system operates as a shared memory machine, and said runtime system further comprises:

a private memory of each thread[[,]] ; the private memory comprising a replica of the shared data directory such that said directory is replicated across all of the threads;

a common directory of shared variables comprising an array of pointers to partitions wherein a calling thread allocates space and inserts a handle in its partition ~~in a common directory of shared variables~~ and any thread can directly access ~~the~~ a control data structure and ~~the~~ a pointer of the array of pointers.

12 and 13 (canceled).

14. (Previously presented) The runtime system of claim 11 wherein each of the shared data structures has affinity to particular threads.

15. (Original) The runtime system of claim 11 wherein the shared data structures comprise shared scalar variables, objects, arrays or pointers.

16. (Original) The runtime system of claim 15 wherein a shared scalar variable is accessed by dereferencing a shared data directory partition for which the shared scalar variable has affinity.

17. (Original) The runtime system of claim 15 wherein a shared array has a shared data directory partition that points to a control structure that points to the shared array.

18. (Previously presented) The runtime system of claim 15 wherein the runtime system allocates a control harness for a shared pointer when the shared pointer is declared by allocating a shared control block and a shared address structure.

19. (Original) The system of claim 15 wherein some of the shared pointers have shared targets and some of the shared pointers have private targets.

20. (Previously presented) The runtime system of claim 11 wherein entries to the shared data directory are allocated by an owning thread or, in a synchronized manner by all threads at the same time.

21. (Original) The runtime system of claim 11 comprising a handle that includes a partition index and a variable index that is used by the threads to access the shared variables.

22. (Original) The runtime system of claim 11 wherein the shared data directory includes a partition that is used to access all statically declared non-scalar variables.

23. (Original) The runtime system of claim 11 wherein each thread uses a mutually exclusive partition of the shared data directory.

24. (Currently amended) A method of providing a scalable runtime system, the method comprising:

running a global address space language program comprising a plurality of program threads;

creating a directory of shared variables, for locating and managing shared objects, the directory containing information concerning data shared by the program threads for use by the threads in accessing the shared data and entries acting as object identifiers, providing a level of address translation between a compiler and the runtime system; and

creating control structures to control allocation and de-allocation of the shared data;

implementing a private memory of each thread, the private memory comprising a replica of the directory of shared variables such that said directory is replicated across all of the threads.

25. (Original) The method of claim 24 wherein creating control structures comprises creating a plurality of control structures wherein each control structure controls the allocation and de-allocation of a particular type of shared data structure.

26. (Original) The method of claim 24 comprising operating the runtime system on a distributed memory machine.

27. (Original) The method of claim 26 wherein each thread contains a private copy of the directory of shared variables and a calling thread allocates an entry in its directory of shared variables and broadcasts an index of the entry to other threads.

28. (Original) The method of claim 26 wherein each thread has a private data control structure with a pointer to a shared memory fraction.

29. (Original) The method of claim 24 comprising operating the runtime system on a shared

memory machine.

30. (Original) The method of claim 24 wherein a calling thread allocates space for a shared variable and inserts a handle in a partition in the directory of shared variables.

31. (Original) The method of claim 29 wherein the control structures are common such that any thread can access the common control structures.